This listing of claims will replace all prior versions, and listings, of claims in the application.

Listing of Claims:

1. (Original) A method for constructing an optimal representation for an input query, the method comprising:

receiving the input query, wherein the input query is an intermediate language representation comprising nodes, each node having a respective node type;

examining the nodes in a left-depth first manner to identify node types for optimization;

tagging nodes corresponding to the identified node types;

moving upward to the next node until the intermediate language representation of the input query has been examined in its entirety;

searching from the top of the intermediate language representation for tagged nodes and identifying associated code patterns to be optimized; and

adjusting the identified code patterns with improved code patterns to form an optimal representation for the input query.

2. (Original) The method of claim 1, wherein the receiving step comprises receiving a semantic intermediate language representation.

3. (Original) The method of claim 2, wherein the semantic representation comprises a graph structure containing nodes.

4. (Original) The method of claim 1, wherein the improved code patterns are generated using one or more translations comprising at least one of constant folding, logical rewrites, path rewrites, loop-invariant code rewrites, tuple rewrites, position rewrites, commutations, inlining and sort elimination.

5. (Original) A computer-readable medium having computer-executable instructions for performing a method for constructing an optimal representation for an input query, the method comprising:

receiving the input query, wherein the input query is an intermediate language representation containing code patterns and nodes, each node having a respective node type;

examining the nodes in a left-depth first manner to identify code patterns and node types which are subjects for optimization;

tagging the identified code patterns until the intermediate language representation of the input query has been examined in its entirety;

searching from the top of the intermediate language representation for tagged code patterns; and

adjusting the tagged code patterns with improved code patterns to form an optimal representation for an input query.

6. (Original) A computer system for generating an optimized representation of an input query comprising:

an input device for receiving an input query;

one or more intermediate language compilers wherein an intermediate language representation containing nodes is generated from the input query; and

an optimizer performing the acts of:

receiving the input query, wherein the input query is an intermediate language representation comprising nodes, each node having a respective node type;

examining the nodes in a left-depth first manner to identify node types for optimization;

tagging nodes corresponding to the identified node types;

moving upward to the next node until the intermediate language representation of the input query has been examined in its entirety;

searching from the top of the intermediate language representation for tagged nodes and identifying associated code patterns to be optimized; and

adjusting the identified code patterns with improved code patterns to form an optimal representation for the input query.

7. (Original) The system of claim 6, further containing a post-optimization processing portion forming query results, comprising:

one or more target generators wherein the optimal representation is transformed into one or more target representations forming a target query;

one or more data sources for querying over; and

one or more execution engines wherein the target query is executed over the one or more data sources to produce the query results.

8. (Original) A computer system for generating an optimized representation of an XML intermediate language representation of one or more of input queries comprising:

one or more of input devices for receiving the one or more input queries;

one or more intermediate language compilers wherein each compiler generates an intermediate language representation of an input query;

an expression accumulator which combines each intermediate language representation into a single XML intermediate language representation; and

an optimizer performing the acts of:

receiving the input query, wherein the input query is an intermediate language representation containing code patterns and nodes, each node having a respective node type;

examining the nodes in a left-depth first manner to identify code patterns and node types which are subjects for optimization;

tagging the identified code patterns until the intermediate language representation of the input query has been examined in its entirety;

searching from the top of the intermediate language representation for tagged code patterns; and

adjusting the tagged code patterns with improved code patterns to form an optimal representation for an input query.

9. (Original) The system of claim 8, wherein the one or more input queries comprise one or more of an XML query and an XML view.

10. (Original) The system of claim 8, further containing a post-optimization process portion forming query results, the system comprising:

one or more target generators wherein the optimized representation is transformed into one or more target representations forming target queries;

one or more data sources for querying over; and

one or more execution engines wherein the target queries are executed over the one or more data sources to produce query results.

11. (Original) A method for constructing an optimal representation for an input query, the method comprising:

receiving the input query, wherein the input query is an intermediate language representation containing nodes, each node having a respective node type;

examining the nodes to inspect code patterns associated with respective node types;

comparing the inspected code patterns using a pattern match algorithm to detect non-optimized code patterns; and

adjusting one or more of the non-optimized code patterns and the inspected code patterns with improved code patterns to form an optimal representation for an input query.

12. (Original) The method of claim 11, wherein the receiving step comprises receiving a semantic intermediate language representation.

13. (Original) The method of claim 12, wherein the semantic representation comprises a graph structure containing nodes.

14. (Original) The method of claim 11, wherein the improved code patterns are generated using one or more translations comprising at least one of constant folding, logical rewrites, path rewrites, loop-invariant code, tuple rewrites, position rewrites, commutations, inlining and sort elimination.

15. (Original) A computer-readable medium having computer-executable instructions for performing a method for constructing an optimal representation for an input query, the method comprising:

receiving the input query, wherein the input query is an intermediate language representation containing nodes, each node having a respective node type;

examining the nodes to inspect code patterns associated with respective node types;

comparing the inspected code patterns using a pattern match algorithm to detect non-optimized code patterns; and

adjusting one or more of the non-optimized code patterns and the inspected code patterns with improved code patterns to form an optimal representation for an input query.

16. (Original) A computer system for generating an optimized representation of an XML intermediate language representation of one or more of input queries comprising:

one or more of input devices for receiving the one or more input queries;

one or more intermediate language compilers wherein each compiler generates an intermediate language representation of an input query;

an expression accumulator which combines each intermediate language representation into a single XML intermediate language representation; and

an optimizer performing the acts of:

receiving the input query, wherein the input query is an intermediate language representation containing nodes, each node having a respective node type;

examining the nodes to inspect code patterns associated with respective node types;

comparing the inspected code patterns using a pattern match algorithm to detect non-optimized code patterns; and

adjusting one or more of the non-optimized code patterns and the inspected code patterns with improved code patterns to form an optimal representation for an input query.